VERIFPAL ANALYSIS

signal.vp

Verifpal 1.6.0 · September 23, 2026

6	3	3
QUERIES	ATTACKS FOUND	NO ATTACK FOUND

This report describes the analysis of the Verifpal model `signal.vp` against an active attacker, with each principal running 2 concurrent sessions. Of its 6 queries, 3 were broken. Every attack reported here is a witness against the model as written. A query reported as holding means that this search found no attack at these parameters, which is not a proof that none exists.

1 signal.vp

Attacker	active
Sessions	2 per principal
Queries	6, with 3 attacks found
Result code	`c1a0c0a1c1a0`
Analysis time	2.2 s

1.1 Verdicts

Table 1: Every query in the model, with the verdict this run reached.

Query	Verdict	Scope	Line
confidentiality? *m1*	attack	witness in §1.3.1	188
authentication? Alice → Bob: *e1*	holds	search exhausted at 2 sessions	189
confidentiality? *m2*	holds	search exhausted at 2 sessions	190
authentication? Bob → Alice: *e2*	attack	witness in §1.3.2	191
confidentiality? *m3*	attack	witness in §1.3.3	192
authentication? Alice → Bob: *e3*	holds	search exhausted at 2 sessions	193

1.2 Protocol

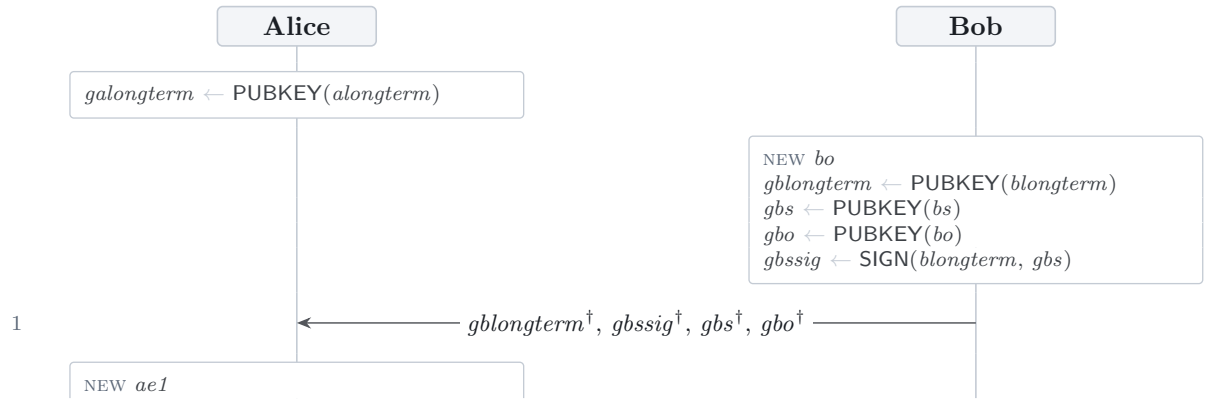

Figure 1, continued

Figure 1, continued

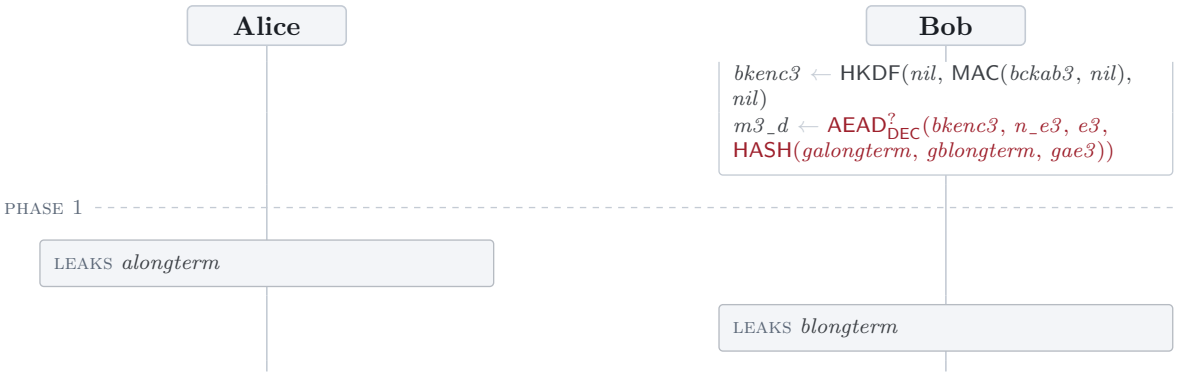

Figure 1: The protocol as written. Guarded values are bracketed; a dagger marks every value an attack below substitutes or replays.

1.3 Attacks

1.3.1 confidentiality? *m1*

m1 (*m1*) is obtained by Attacker.

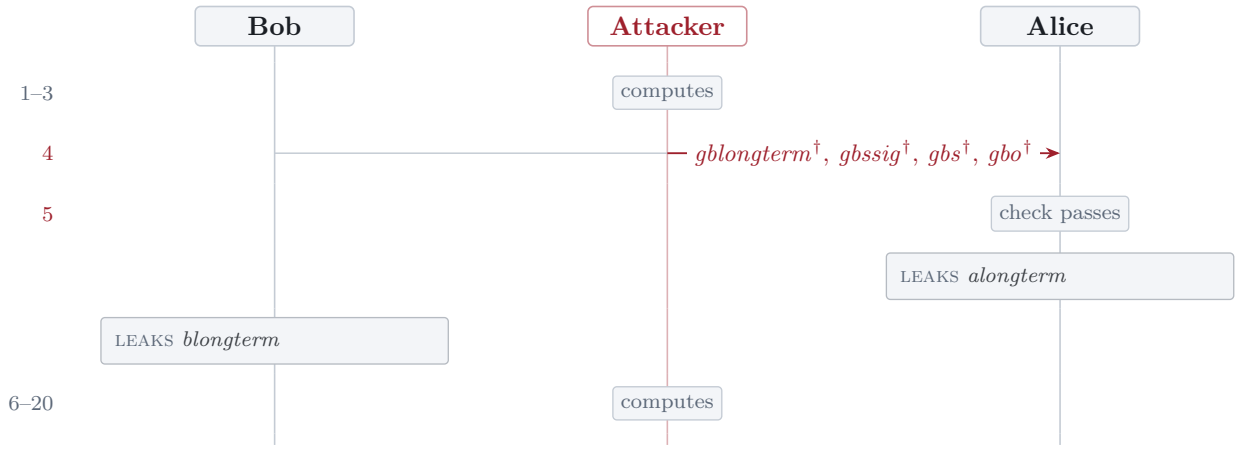

Figure 2: How the attacker breaks confidentiality? *m1*. Substituted values carry a dagger.

1. Attacker constructs PUBKEY(*nil*).
2. Attacker observes *gblongterm* on the wire.
3. Attacker constructs SIGN(*nil*, *gblongterm*).
4. Attacker replaces *gblongterm*, *gbssig*, *gbs*, *gbo* (sent by Bob to Alice) with PUBKEY(*nil*), SIGN(*nil*, *gblongterm*), *gblongterm*, *gblongterm*. (*gblongterm* was PUBKEY(*blongterm*); *gbssig* was SIGN(*blongterm*, *gbs*); *gbs* was PUBKEY(*bs*); *gbo* was PUBKEY(*bo*)) Bob → Alice
5. Alice's SIGNVERIF?(PUBKEY(*nil*), PUBKEY(*blongterm*), SIGN(*nil*, PUBKEY(*blongterm*))) passes — the attacker controls one of its inputs.
6. Attacker observes *e1* on the wire, where it is AEAD$_{\text{ENC}}$(*akenc1*, *n_e1*, *m1*, HASH(*galongterm*, PUBKEY(*nil*), *gae2*)).
7. Attacker is handed *alongterm* by a leaks declaration in Alice#2.
8. Attacker constructs DH$_{\text{KEX}}$(*gblongterm*, *alongterm*).
9. Attacker observes *gae1* on the wire.
10. Attacker constructs DH$_{\text{KEX}}$(*gae1*, *nil*).
11. Attacker is handed *blongterm* by a leaks declaration in Bob#2.
12. Attacker constructs DH$_{\text{KEX}}$(*gae1*, *blongterm*).
13. Attacker constructs *amaster*, where it is HASH(DH$_{\text{KEX}}$(*gblongterm*, *alongterm*), DH$_{\text{KEX}}$(*gae1*, *nil*), DH$_{\text{KEX}}$(*gae1*, *blongterm*), DH$_{\text{KEX}}$(*gae1*, *blongterm*)).
14. Attacker observes *gae2* on the wire.
15. Attacker constructs *akshared1*, where it is DH$_{\text{KEX}}$(*gae2*, *blongterm*).
16. Attacker constructs *ackab1*, where it is HKDF(*amaster*, *akshared1*, *nil*)$_2$.
17. Attacker constructs MAC(*ackab1*, *nil*).

18 Attacker constructs *akenc1*, where it is HKDF(*nil*, MAC(*ackab1*, *nil*), *nil*)$_1$.
19 Attacker observes *n_e1* on the wire.
20 Attacker opens *e1* with *akenc1*, *n_e1*, obtaining *m1*.

1.3.2 authentication? Bob → Alice: *e2*

e2 (AEAD$_{\mathsf{ENC}}$(*akenc2*, *nil*, *nil*, HASH(*galongterm*, PUBKEY(*nil*), PUBKEY(*nil*)))), sent by Attacker and not by Bob, is successfully used in AEAD$^{?}_{\mathsf{DEC}}$(*akenc2*, *n_e2*, *e2*, HASH(*galongterm*, *gblongterm*, *gbe*)) within Alice's state.

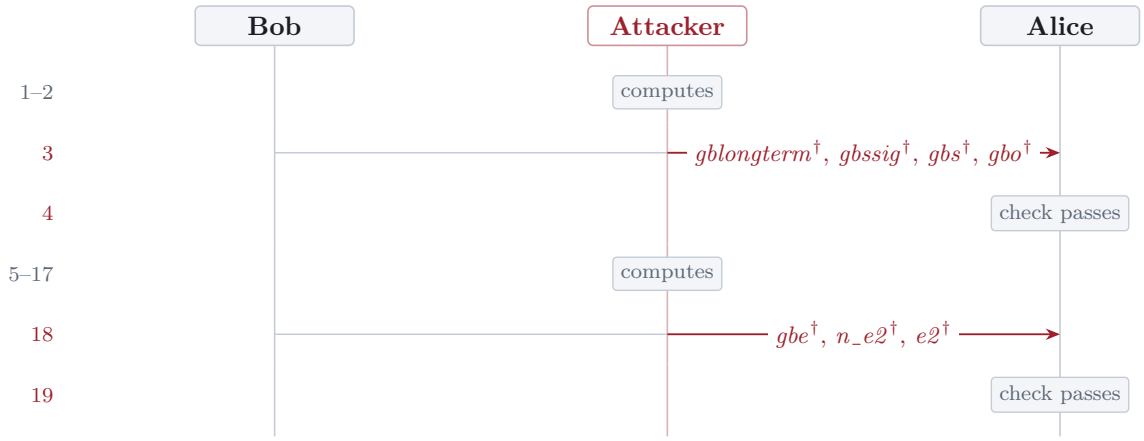

Figure 3: How the attacker breaks authentication? Bob → Alice: *e2*. Substituted values carry a dagger.

1 Attacker constructs PUBKEY(*nil*).
2 Attacker constructs SIGN(*nil*, PUBKEY(*nil*)).
3 Attacker replaces *gblongterm*, *gbssig*, *gbs*, *gbo* (sent by Bob to Alice) with PUBKEY(*nil*), SIGN(*nil*, PUBKEY(*nil*)), PUBKEY(*nil*), PUBKEY(*nil*). (*gblongterm* was PUBKEY(*blongterm*); *gbssig* was SIGN(*blongterm*, *gbs*); *gbs* was PUBKEY(*bs*); *gbo* was PUBKEY(*bo*)) Bob → Alice
4 Alice's SIGNVERIF$^{?}$(PUBKEY(*nil*), PUBKEY(*nil*), SIGN(*nil*, PUBKEY(*nil*))) passes — the attacker controls one of its inputs.
5 Attacker observes *galongterm* on the wire.
6 Attacker constructs DH$_{\mathsf{KEX}}$(*galongterm*, *nil*).
7 Attacker observes *gae1* on the wire.
8 Attacker constructs DH$_{\mathsf{KEX}}$(*gae1*, *nil*).
9 Attacker constructs *amaster*, where it is HASH(DH$_{\mathsf{KEX}}$(*galongterm*, *nil*), DH$_{\mathsf{KEX}}$(*gae1*, *nil*), DH$_{\mathsf{KEX}}$(*gae1*, *nil*), DH$_{\mathsf{KEX}}$(*gae1*, *nil*)).
10 Attacker observes *gae2* on the wire.
11 Attacker constructs *akshared1*, where it is DH$_{\mathsf{KEX}}$(*gae2*, *nil*).
12 Attacker constructs *arkab1*, where it is HKDF(*amaster*, *akshared1*, *nil*)$_1$.
13 Attacker constructs *ackba2*, where it is HKDF(*arkab1*, *akshared1*, *nil*)$_2$.
14 Attacker constructs MAC(*ackba2*, *nil*).
15 Attacker constructs *akenc2*, where it is HKDF(*nil*, MAC(*ackba2*, *nil*), *nil*)$_1$.
16 Attacker constructs HASH(*galongterm*, PUBKEY(*nil*), PUBKEY(*nil*)).
17 Attacker constructs AEAD$_{\mathsf{ENC}}$(*akenc2*, *nil*, *nil*, HASH(*galongterm*, PUBKEY(*nil*), PUBKEY(*nil*))).
18 Attacker replaces *gbe*, *n_e2*, *e2* (sent by Bob to Alice) with PUBKEY(*nil*), *nil*, AEAD$_{\mathsf{ENC}}$(*akenc2*, *nil*, *nil*, HASH(*galongterm*, PUBKEY(*nil*), PUBKEY(*nil*))). (*gbe* was PUBKEY(*be*); *e2* was AEAD$_{\mathsf{ENC}}$(*akenc2*, *n_e2*, *m2*, HASH(*galongterm*, *gblongterm*, *gbe*))) Bob → Alice
19 Alice's AEAD$^{?}_{\mathsf{DEC}}$(*akenc2*, *nil*, AEAD$_{\mathsf{ENC}}$(*akenc2*, *nil*, *nil*, HASH(*galongterm*, PUBKEY(*nil*), PUBKEY(*nil*))), HASH(*galongterm*, PUBKEY(*nil*), PUBKEY(*nil*))) passes — the attacker controls one of its inputs.

1.3.3 confidentiality? *m3*

m3 (*m3*) is obtained by Attacker.

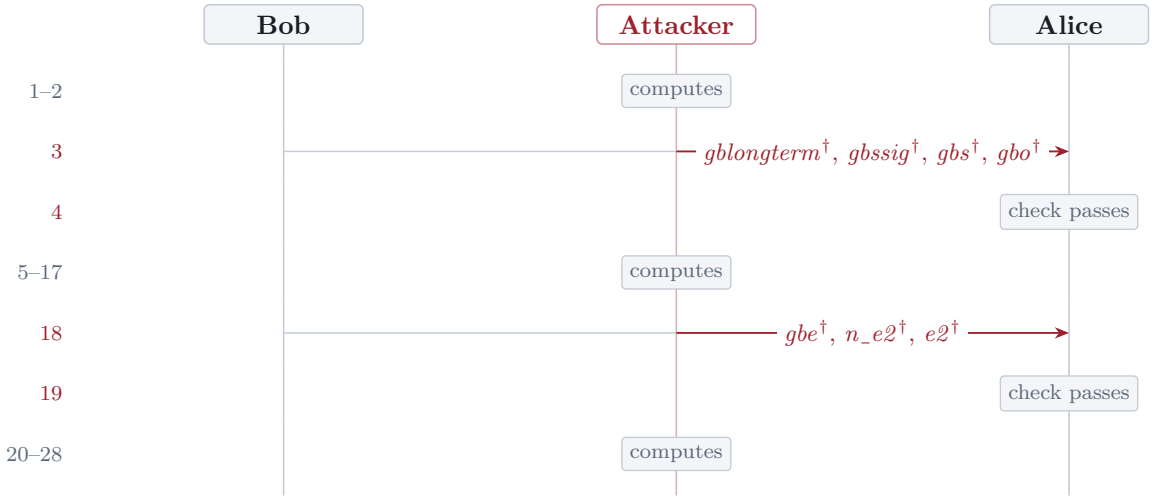

Figure 4: How the attacker breaks confidentiality? *m3*. Substituted values carry a dagger.

1. Attacker constructs PUBKEY(*nil*).
2. Attacker constructs SIGN(*nil*, PUBKEY(*nil*)).
3. Attacker replaces *gblongterm*, *gbssig*, *gbs*, *gbo* (sent by Bob to Alice) with PUBKEY(*nil*), SIGN(*nil*, PUBKEY(*nil*)), PUBKEY(*nil*), PUBKEY(*nil*). (*gblongterm* was PUBKEY(*blongterm*); *gbssig* was SIGN(*blongterm*, *gbs*); *gbs* was PUBKEY(*bs*); *gbo* was PUBKEY(*bo*)) Bob → Alice
4. Alice's $\text{SIGNVERIF}^{?}$(PUBKEY(*nil*), PUBKEY(*nil*), SIGN(*nil*, PUBKEY(*nil*))) passes — the attacker controls one of its inputs.
5. Attacker observes *galongterm* on the wire.
6. Attacker constructs DH_{KEX}(*galongterm*, *nil*).
7. Attacker observes *gae1* on the wire.
8. Attacker constructs DH_{KEX}(*gae1*, *nil*).
9. Attacker constructs *amaster*, where it is HASH(DH_{KEX}(*galongterm*, *nil*), DH_{KEX}(*gae1*, *nil*), DH_{KEX}(*gae1*, *nil*), DH_{KEX}(*gae1*, *nil*)).
10. Attacker observes *gae2* on the wire.
11. Attacker constructs *akshared1*, where it is DH_{KEX}(*gae2*, *nil*).
12. Attacker constructs *arkab1*, where it is $\text{HKDF}(\textit{amaster}, \textit{akshared1}, \textit{nil})_1$.
13. Attacker constructs *ackba2*, where it is $\text{HKDF}(\textit{arkab1}, \textit{akshared1}, \textit{nil})_2$.
14. Attacker constructs MAC(*ackba2*, *nil*).
15. Attacker constructs *akenc2*, where it is $\text{HKDF}(\textit{nil}, \text{MAC}(\textit{ackba2}, \textit{nil}), \textit{nil})_1$.
16. Attacker constructs HASH(*galongterm*, PUBKEY(*nil*), PUBKEY(*nil*)).
17. Attacker constructs AEAD_{ENC}(*akenc2*, *nil*, *nil*, HASH(*galongterm*, PUBKEY(*nil*), PUBKEY(*nil*))).
18. Attacker replaces *gbe*, *n_e2*, *e2* (sent by Bob to Alice) with PUBKEY(*nil*), *nil*, AEAD_{ENC}(*akenc2*, *nil*, *nil*, HASH(*galongterm*, PUBKEY(*nil*), PUBKEY(*nil*))). (*gbe* was PUBKEY(*be*); *e2* was AEAD_{ENC}(*akenc2*, *n_e2*, *m2*, HASH(*galongterm*, *gblongterm*, *gbe*))) Bob → Alice
19. Alice's $\text{AEAD}^{?}_{\text{DEC}}$(*akenc2*, *nil*, AEAD_{ENC}(*akenc2*, *nil*, *nil*, HASH(*galongterm*, PUBKEY(*nil*), PUBKEY(*nil*))), HASH(*galongterm*, PUBKEY(*nil*), PUBKEY(*nil*))) passes — the attacker controls one of its inputs.
20. Attacker observes *e3* on the wire, where it is AEAD_{ENC}(*akenc3*, *n_e3*, *m3*, HASH(*galongterm*, PUBKEY(*nil*), *gae3*)).
21. Attacker constructs *arkba2*, where it is $\text{HKDF}(\textit{arkab1}, \textit{akshared1}, \textit{nil})_1$.
22. Attacker observes *gae3* on the wire.
23. Attacker constructs *akshared3*, where it is DH_{KEX}(*gae3*, *nil*).
24. Attacker constructs *ackab3*, where it is $\text{HKDF}(\textit{arkba2}, \textit{akshared3}, \textit{nil})_2$.
25. Attacker constructs MAC(*ackab3*, *nil*).
26. Attacker constructs *akenc3*, where it is $\text{HKDF}(\textit{nil}, \text{MAC}(\textit{ackab3}, \textit{nil}), \textit{nil})_1$.
27. Attacker observes *n_e3* on the wire.
28. Attacker opens *e3* with *akenc3*, *n_e3*, obtaining *m3*.

1.4 Scope and assumptions

Every verdict above was reached against an active attacker, with each principal running 2 concurrent sessions, over exactly the model as written. An attack is a witness and stands on its own. A query reported as holding says only that this search found no attack at those parameters: the search space this engine defines was explored, which is never the space of all attacks.

- Per-session values and principals carry the suffix #2.

A On soundness

Verifpal is sound but incomplete. Every attack shown here is a genuine attack on the model as written; a query reported as holding means no attack was found within the search this run performed, which is never a proof that none exists.

B Model source

B.1 `signal.vp`

```
// SPDX-FileCopyrightText: © 2019-2026 Nadim Kobeissi <nadim@symbolic.software>
// SPDX-License-Identifier: GPL-3.0-only
//
// Expected: c0a0c0a0c0a0 at one session and at two.
//
// The Signal protocol: an X3DH handshake followed by three messages of the
// Double Ratchet, following the two Signal specifications (X3DH revision 1,
// The Double Ratchet Algorithm revision 4). Phase 1 then leaks BOTH parties'
// long-term identity keys, and every query still holds (c0a0c0a0c0a0) - the
// model demonstrates the two properties Signal is famous for:
//
//   Forward secrecy: message keys descend from ephemeral Diffie-Hellman
//   shares (via the X3DH secret and each ratchet step), so a later
//   compromise of identity keys opens no recorded ciphertext.
//
//   Authentication under compromise: the leak happens after phase 0, and
//   phases model "later" - within the conversation, every ciphertext is
//   bound to the sender's ratchet state, which the attacker cannot enter
//   without a private ephemeral it never gets.
//
// X3DH in one paragraph: Bob publishes an identity key IK_B (gblongterm), a
// signed prekey SPK_B (gbs, signature gbssig) and a one-time prekey OPK_B
// (gbo). Alice verifies the prekey signature, generates an ephemeral EK_A
// (ae1), and derives the session secret SK (amaster) from four DH values:
//
//   DH1 = DH(IK_A, SPK_B)   - authenticates Alice to Bob
//   DH2 = DH(EK_A, IK_B)    - authenticates Bob to Alice
//   DH3 = DH(EK_A, SPK_B)   - forward secrecy from Alice's side
//   DH4 = DH(EK_A, OPK_B)   - forward secrecy from Bob's side
//   SK  = KDF(DH1 || DH2 || DH3 || DH4)
//
// No single compromised value breaks all four at once. SK then seeds the
// Double Ratchet: it is the initial root key, and Bob's signed prekey
// doubles as his initial ratchet key pair (Double Ratchet §7.1). Every
// ratchet step below is the spec's key schedule:
//
//   RK', CK = KDF_RK(RK, DH(own ratchet key, peer ratchet key))
//             = HKDF(salt: RK, ikm: the fresh DH share)
//   mk      = KDF_CK(CK) - an HMAC keyed by the chain key: MAC(ck, nil)
//
// and the AEAD key is expanded from mk the way the spec's recommended
// ENCRYPT does internally: a zero-salt HKDF of mk (§7.2). This conversation
```

```
// alternates strictly, so every send answers a freshly received ratchet
// public key with a full DH ratchet step; each chain key therefore yields
// exactly one message key, and KDF_CK's next-chain-key output is elided.
// New ratchet keys are generated at send time - the deferred variant of
// §8.5. The associated data models CONCAT(AD, header): the X3DH identity
// binding AD = Encode(IK_A) || Encode(IK_B), in that fixed order for both
// directions, plus the header's ratchet public key (its message counters
// carry no secrets and are elided).
//
// Modeling notes: the identity public keys travel guarded …([]) - Signal
// users verify safety numbers out-of-band; try removing a guard and watch
// the queries fail. The prekey signature is verified with a checked
// SIGNVERIF before Alice touches gbs, exactly as X3DH §3.3 orders the
// steps. bs is `knows private` while bo is `generates`: under Verifpal's
// parallel sessions, the medium-term signed prekey is shared across
// sessions while each session gets a fresh one-time prekey, matching the
// two lifetimes in the spec. See pqxdh.vp for Signal's post-quantum
// successor to this handshake, and signal_twelve.vp for this same model
// extended to twelve ratchet messages.

attacker[active]

principal Alice[
    // Alice's long-term identity key pair, IK_A.
    knows private alongterm
    galongterm = PUBKEY(alongterm)
]

principal Bob[
    // blongterm: the identity key IK_B. bs: the signed prekey SPK_B,
    // uploaded to the server in advance so Alice can start a session while
    // Bob is offline - it doubles as Bob's initial ratchet key pair.
    knows private blongterm, bs
    // bo: a one-time prekey OPK_B, fresh per session.
    generates bo
    gblongterm = PUBKEY(blongterm)
    gbs = PUBKEY(bs)
    gbo = PUBKEY(bo)
    // Bob signs his prekey with his identity key: this is what stops an
    // attacker from handing Alice a prekey of its own.
    gbssig = SIGN(blongterm, gbs)
]

// Bob's "prekey bundle", as fetched from the Signal server. The identity
// key is guarded - its authenticity is anchored out-of-band (safety
// numbers) - while the rest is protected by the signature, not the wire.
Bob → Alice: gblongterm, gbssig, gbs, gbo

principal Alice[
    // Checked signature verification, before the prekey is used anywhere:
    // Alice aborts rather than run X3DH against an unsigned prekey.
    _ = SIGNVERIF(gblongterm, gbs, gbssig)?
    generates ae1
    gae1 = PUBKEY(ae1)
    // The four X3DH shared secrets, combined into the session secret SK.
    amaster = HASH(DH_KEX(gbs, alongterm), DH_KEX(gblongterm, ae1), DH_KEX(gbs, ae1), DH_KEX(gbo,
     ae1))
]

principal Alice[
    generates m1, ae2
    // ae2 is Alice's initial ratchet key; Bob's half is his signed prekey.
    gae2 = PUBKEY(ae2)
    akshared1 = DH_KEX(gbs, ae2)
    // Ratchet initialization, one root-KDF step: SK is the initial root
```

```
    // key, mixed with the first ratchet DH share - KDF_RK(SK, DH(ae2, SPK_B)).
    arkab1, ackab1 = HKDF(amaster, akshared1, nil)
    // Message key derivation: KDF_CK is an HMAC keyed by the chain key,
    // and ENCRYPT expands it into the AEAD key with a zero-salt HKDF.
    akenc1 = HKDF(nil, MAC(ackab1, nil), nil)
    // The associated data binds both identities - AD = IK_A || IK_B, the
    // same order in both directions - and the header's ratchet key, so
    // replaying the ciphertext in another context fails AEAD_DEC.
    generates n_e1
    e1 = AEAD_ENC(akenc1, n_e1, m1, HASH(galongterm, gblongterm, gae2))
]

Alice → Bob: [galongterm], gae1, gae2, n_e1, e1

principal Bob[
    // Bob mirrors X3DH from his side: same four DH values, opposite
    // private/public halves - DH_KEX(PUBKEY(a), b) = DH_KEX(PUBKEY(b), a).
    bmaster = HASH(DH_KEX(galongterm, bs), DH_KEX(gae1, blongterm), DH_KEX(gae1, bs), DH_KEX(gae1,
     bo))
]

principal Bob[
    // The receiving half of the same ratchet step.
    bkshared1 = DH_KEX(gae2, bs)
    brkab1, bckab1 = HKDF(bmaster, bkshared1, nil)
    bkenc1 = HKDF(nil, MAC(bckab1, nil), nil)
    // Checked decryption: a forged or replayed e1 halts Bob here.
    m1_d = AEAD_DEC(bkenc1, n_e1, e1, HASH(galongterm, gblongterm, gae2))?
]

// Message 2: Bob replies, advancing the DH ratchet with his own fresh
// ephemeral be, generated at send time (the deferred variant of §8.5).
// Note the root-key chaining: each KDF_RK takes the previous root key as
// its salt, so history is folded into every future key.
principal Bob[
    generates m2, be
    gbe = PUBKEY(be)
    bkshared2 = DH_KEX(gae2, be)
    brkba2, bckba2 = HKDF(brkab1, bkshared2, nil)
    bkenc2 = HKDF(nil, MAC(bckba2, nil), nil)
    generates n_e2
    e2 = AEAD_ENC(bkenc2, n_e2, m2, HASH(galongterm, gblongterm, gbe))
]

Bob → Alice: gbe, n_e2, e2

principal Alice[
    akshared2 = DH_KEX(gbe, ae2)
    arkba2, ackba2 = HKDF(arkab1, akshared2, nil)
    akenc2 = HKDF(nil, MAC(ackba2, nil), nil)
    m2_d = AEAD_DEC(akenc2, n_e2, e2, HASH(galongterm, gblongterm, gbe))?
]

// Message 3: back to Alice - another full ratchet step.
principal Alice[
    generates m3, ae3
    gae3 = PUBKEY(ae3)
    akshared3 = DH_KEX(gbe, ae3)
    arkab3, ackab3 = HKDF(arkba2, akshared3, nil)
    akenc3 = HKDF(nil, MAC(ackab3, nil), nil)
    generates n_e3
    e3 = AEAD_ENC(akenc3, n_e3, m3, HASH(galongterm, gblongterm, gae3))
]

Alice → Bob: gae3, n_e3, e3
```

```
171
172 principal Bob[
173     bkshared3 = DH_KEX(gae3, be)
174     brkab3, bckab3 = HKDF(brkba2, bkshared3, nil)
175     bkenc3 = HKDF(nil, MAC(bckab3, nil), nil)
176     m3_d = AEAD_DEC(bkenc3, n_e3, e3, HASH(galongterm, gblongterm, gae3))?
177 ]
178
179 phase[1]
180
181 // Both identity keys fall - a full long-term compromise of both parties,
182 // after the conversation. The ephemerals (ae1, ae2, ae3, be, bs, bo) do
183 // not leak; they are what forward secrecy stands on.
184 principal Alice[leaks alongterm]
185 principal Bob[leaks blongterm]
186
187 queries[
188     confidentiality? m1
189     authentication? Alice → Bob: e1
190     confidentiality? m2
191     authentication? Bob → Alice: e2
192     confidentiality? m3
193     authentication? Alice → Bob: e3
194 ]
```